Vol. III, No. 5
May/June 1984

SATN®

Software Arts Technical Notes from the creators of VisiCalc® and TK!Solver®

To Our Readers

Beginning with this issue, SATN will contain articles about all Software Arts products. Right now that means VisiCalc®, VisiCalc Advanced Version, and TK!Solver®. With expanded coverage will come expanded size, beginning with the September/October issue.

CONTENTS	FEATURING	PAGE
Decisions, Decisions We build a decision analysis model, which you can develop with any version of the VisiCalc program. Using various built-in functions, the program can weigh your choices and identify which selection is best for you. In this model we also show you how to build a "switch."	@IF @INT @MAX @DOTPROD @LOOKUP @LABEL @LCHOOSE /AH	2
Windows on the Worksheet When you split your screen into two windows, you can set a different format, column width, and title setting in each window.	/W /G /T	6
Drawing a Blank The VisiCalc and TK!Solver programs treat blanks slightly differently. In this article, we examine how that difference may affect you, particularly when dealing with lists and certain functions that have been built into the TK!Solver program.	/L! POLY NPV	8
From Our Readers A reader suggests a way to expand an @LOOKUP range without using forward references.		10
Related Products Once again we list some of the products that enhance the VisiCalc program.		10

DECISIONS, DECISIONS

Choosing between two or more desirable options can often be difficult, whether the choice involves deciding which house to buy, which job offer to accept, or anything else. With the VisiCalc® program, you can develop a model that will help you make decisions more easily. By using VisiCalc Advanced Version's @DOTPROD, @LCHOOSE, @LABEL, and other built-in functions, you can calculate and identify which option is best for you. With other versions of the VisiCalc program, you can develop a similar model with the @INT and @MAX functions.

In this article, we'll build a decision analysis model that determines which of four cars to buy. The decision is based on eight factors used to compare the four cars and the weight assigned to each factor. You can easily adapt this model to help solve any of your decision-making problems.

Analyzing the Choices

Before you make any decisions, you should always take a close look at your options and determine the criteria upon which you will base your choice. For instance, in choosing a restaurant, you might be influenced by the price, service, location, atmosphere, and quality of food. When buying a personal computer, in addition to the cost, you might consider the expandability and memory capacity; the type of disk drive, operating system, and keyboard; the variety of software available; and the reputation of the manufacturer.

After you have determined which factors are important in your decision making, you should rank them according to their relative significance. Among the factors that affect your choice of a restaurant, perhaps the quality of food is most important to you, closely followed by the service, atmosphere, price, and then the location. For a personal computer, you might be most concerned with the memory capacity and then the software availability, with the other factors being given less weight.

For our model, we rank the following eight factors according to our priorities for buying a new car: style, comfort, acceleration, safety record, availability, mileage, service record, and trunk space.

Building the Model

To begin building the decision analysis model, set the global column width to 12 (**/GC12**) and the global order of recalculation to row (**/GOR**). Next, create a continuous dashed line across columns A through F in rows 3 and 12. Enter the other labels in A1 through A13, as shown in Figure 1.

To right justify the rest of the labels in rows 1 and 2, enter:

```
>B2:/FR/R:B1...F1
```

The car prices in C2 through F2 should be displayed in the dollars-and-cents format, so enter:

```
>C2:/F$/R:D2...F2
```

Now enter the labels and values in columns B through F, rows 1 and 2, as shown in Figure 1. Be sure to begin the car names with a quotation mark (") so that they are entered as labels, not values.

In the Weighting Factors column, we ranked the Decision Factors with the numbers 1 through 8, according

How to Use SATN

The VisiCalc program is available on many different personal computers. These machines may have slightly different keyboards. To reduce confusion, SATN will use the symbol ® and the word RETURN synonymously to represent the keys known as ENTER, ENDLINE, and RETURN. The symbol ∧ is used to represent the exponentiation operator.

To make VisiCalc documentation more readable, the program was designed so that the colon (:) has the same effect as RETURN in some commands, as when ending the > command and to separate the various elements of the replicate command.

In examples of what to type, type exactly what you see and press RETURN at the end of every line. On some computers, you may find it necessary to add parentheses to some of the expressions included in the text. For example, 3 + 4*5 may become (3 + 4) *5.

Throughout SATN, one typeface represents both VisiCalc input and output (what you should type and what you will see on the screen). The typeface looks like this:

```
This line is typeset in OCR-B.
So is this: +A1+B1*3.25
```

Characters you should type in are printed in rust.

To allow you to type the formulas directly into VisiCalc, the examples use specific coordinates, like A1, rather than the names of the value they represent. When actually typing a formula, you can use the arrows to point rather than using the > command.

SATN is published bimonthly by Software Arts Products Corp., 27 Mica Lane, Wellesley, MA 02181. Address all editorial correspondence to the Editorial Department, SATN, P.O. Box 99, Newton, MA 02162. Requests for special permission or bulk orders should be addressed to the publisher.

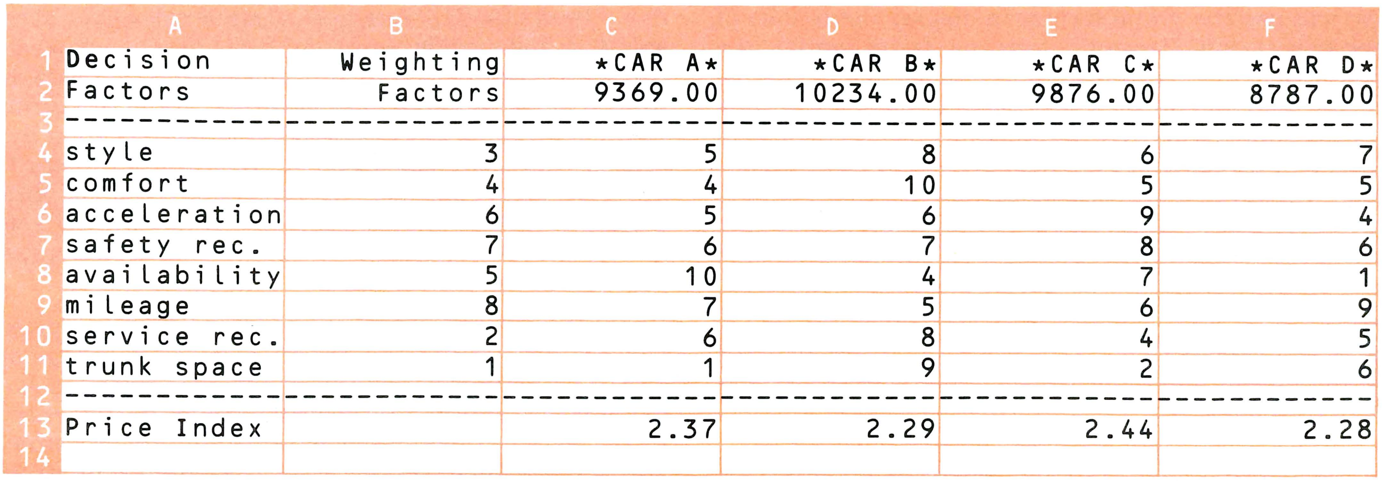
Domestic subscriptions: 6 issues $30.00, 12 issues $50.00; outside U.S.A., 6 issues $50.00, 12 issues $90.00. Single-copy price, $6.00. Back-issue price, $7.00. Send subscriptions, change of address, and fulfillment questions to SATN Subscriptions, P.O. Box 97, Newton, MA 02162.

Copyright © 1984 Software Arts Products Corp. All rights reserved. This periodical may not be copied or reproduced in any fashion either in whole or in part without the prior written consent of Software Arts Products Corp.

Software Arts Products Corp. makes no warranty, express or implied, with respect to the quality, accuracy, or freedom from error of the information contained in SATN, and specifically disclaims all liability for any damages resulting from the use of such information.

Unless specifically stated otherwise in any article in this issue of SATN, Software Arts Products Corp. has not examined any products and/or publications identified in SATN to determine their accuracy and/or ability to perform as advertised. Mention of a product and/or publication in SATN does not constitute an endorsement by Software Arts Products Corp. of any such product or publication.

Recognition in SATN of the existence of any bug or problem in any product distributed by Software Arts, Inc. does not constitute, and should not be construed as, an acknowledgment by Software Arts, Inc. of any deficiency or as a commitment by Software Arts, Inc. to correct the bug or problem.



	A	B	C	D	E	F
1	Decision	Weighting	*CAR A*	*CAR B*	*CAR C*	*CAR D*
2	Factors	Factors	9369.00	10234.00	9876.00	8787.00
3	----------	----------	----------	----------	----------	----------
4	style	3	5	8	6	7
5	comfort	4	4	10	5	5
6	acceleration	6	5	6	9	4
7	safety rec.	7	6	7	8	6
8	availability	5	10	4	7	1
9	mileage	8	7	5	6	9
10	service rec.	2	6	8	4	5
11	trunk space	1	1	9	2	6
12	----------	----------	----------	----------	----------	----------
13	Price Index		2.37	2.29	2.44	2.28
14						

FIGURE 1. *In column B, the eight Decision Factors are ranked according to our priorities for a new car. In columns C through F, the cars are scored on the basis of how they rate on each of the eight factors.*

to their level of importance. Because your priorities may differ from ours, you can rank the factors in any order you like. Just be sure that you assign 1 to the least important factor and 8 to the most important factor. For instance, if you do not drive very much, you might give mileage a ranking of 2 or 3. If you have a large family and intend to use the car for family vacations, trunk space would be an important factor for you. So you might give it a ranking of 6.

In columns C through F, rows 4 through 11, we rated each of the four cars based on the eight Decision Factors. The possible scores for each factor range from 1 to 10, with 1 being the worst and 10 the best. For example, a car dealer has CAR A in his showroom and it can be delivered immediately. So CAR A rates a 10 for availability. Similarly, CAR D averages 48 miles per gallon on the highway and 36 in the city. Therefore, it rates a 9 for mileage.

Now enter the values in C4 through F11, as shown in Figure 1. Or, if you wish, you can enter other values according to how you would rate four different cars.

The values in the Price Index row provide a basis for comparing the four cars. They represent an evaluation of each car's features relative to its price.

Before entering the expression that produces the Price Index for each car, format cells C13 through F13 so that the values are displayed with two decimal places. The program will still carry the values in memory with 11 or 12 significant digits. Enter:

```
>C13:/F$/R:D13...F13
```

To produce the Price Index, multiply each Weighting Factor by the corresponding Decision Factor rating, add the eight resulting numbers, and then divide the sum by the car price. Next, multiply the total by 100 to make the index number appear more meaningful. If you are not using the Advanced Version of the program, enter:

```
>C13:(B4*C4)+(B5*C5)+(B6*C6)+(B7*C7)+
  (B8*C8)+(B9*C9)+(B10*C10)+
   (B11*C11)/C2*100
```

Now replicate the expression:

```
>C13:/R:D13...F13:NRNRNRNRNRNRNRNRR
```

With the Advanced Version of the program, you can use the @DOTPROD function to calculate the sum of each Weighting Factor multiplied by the corresponding Decision Factor. To generate the Price Index, enter:

AVO
```
>C13:@DOTPROD(B4...B11,C4...
 C11)/C2*100
```

Replicate the contents of C13 into D13 through F13:

AVO
```
>C13:/R:D13...F13:NNRRR
```

Making the Decision

By simply looking at the four numbers in the Price Index row, you can determine which car has the highest index. However, if you were evaluating numerous options, it would be difficult to glance at them all at once and immediately find the highest number. Therefore, make the model more complete by entering an expression that indicates which car has the highest Price Index.

This expression should function as an on/off switch. The value should be 1 (on) if the corresponding Price Index is the highest of the indices and 0 (off) if the index is not the highest.

To develop this expression, use the @INT and @MAX functions. Divide each Price Index by the maximum of the four indices and then take the integer portion of the resulting value. If an index equals the maximum, it is divided by itself and the integer portion is 1. If an index is less than the maximum, it is divided by a number that is greater than itself and the integer portion is 0.

Now enter:

```
>C14:@INT(C13/@MAX(C13...F13))
```

Replicate the expression to produce the on/off switch for the other three cars:

```
>C14:/R:D14...F14:RNN
```

If you are using the Advanced Version of the program, skip ahead to the section entitled *Making the Decision with VAV.*

We will use the graph format to display the results of the expression. Presently with the graph format, the on/off switch will display either a blank cell or one asterisk. We can edit the expression to make the results appear more clearly. By multiplying it by 11, instead of cells C14 through F14 having only one or no asterisks, they will be filled with asterisks or be empty, depending on which car has the highest Price Index. (See Figure 2.) And if two or more cars are tied for the maximum value, asterisks will fill each of the appropriate cells.

Now edit the expression at C14 to read: @INT(C13/@MAX(C13...F13))*11. Then use the graph format and replicate the edited expression into D14 through F14:

```
>C14:/F*/R:D14...F14:RNN
```

Making the Decision with VAV

The following paragraphs are for users of the Advanced Version of the program. If you are not using that version, go to the section entitled *Changing Priorities*.

By using the Advanced Version's @LABEL and @LCHOOSE functions along with a lookup table, you can create an expression that will return the label of the car you should buy or tell you if two or more cars are tied for the highest Price Index. Though you can accomplish this task without developing a lookup table, the expression needed to do so would be extremely long and complicated. By using a series of simpler expressions with a lookup table, you can keep your model more manageable.

The lookup table finds the column containing the car with the highest Price Index. It searches the values in row 14 and then returns a value from the adjacent range (row 15).

Now enter the values for the adjacent range:

```
AVO   >C15:1
AVO   >D15:2
AVO   >E15:3
AVO   >F15:4
```

For the purposes of this table, CAR A is 1, CAR B is 2, CAR C is 3, and CAR D is 4. (See row 15 in Figure 3.) With this information, the program can choose the label at the top of the appropriate column to indicate which car you should buy.

With the lookup table set up, you can enter in A15 an expression that will return the desired value: the number corresponding to the car with the highest Price Index. The @LOOKUP function's search value should be less than 1 since it is compared to successive values in the search range, of which the maximum value is 1, until a value is found that is **larger** than the search value. If no value in the lookup table is larger than the search value, the last value in the table is returned. Since the last value may not correspond to the car with the highest Price Index, using a search value of 1 or greater may return an incorrect answer. Therefore, use a search value of .5.

Begin the @LOOKUP search range at B14 (which is evaluated as 0 since it is blank), and end the range at F14. The boundaries of this range are important because if the search value is smaller than the first entry in the lookup table, @NA is returned. For example, with a range beginning at C14 and a search value of .5, if the first entry (at C14) were 1, the function would return @NA. With the search range beginning at B14, that will not happen.

For the final part of the expression, add 1 to the result of the @LOOKUP statement. The reason for doing so is that when the function finds a match position (the first value that is greater than the search value), it "backs up" one entry and returns the corresponding entry in the adjacent range. The value that we want returned, however, is the entry that corresponds to the match position, since this is where the highest Price Index is. Because the values in the adjacent range are 1 to 4, the the function always returns a value that is 1 less than the value at the match position. So by adding 1 to the result of the @LOOKUP statement, we get the number that corresponds to the car with the highest Price Index.

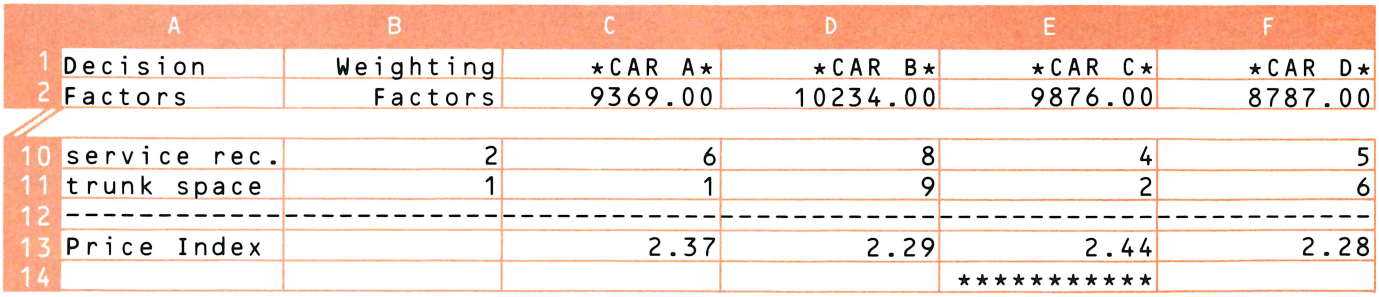
Now combine the parts of this expression and enter:

```
AVO   >A15:@LOOKUP(.5,B14...F14)+1
```

In D16, enter an expression that will either return the label of the car you should buy or tell you that two or more cars are tied for the highest Price Index.

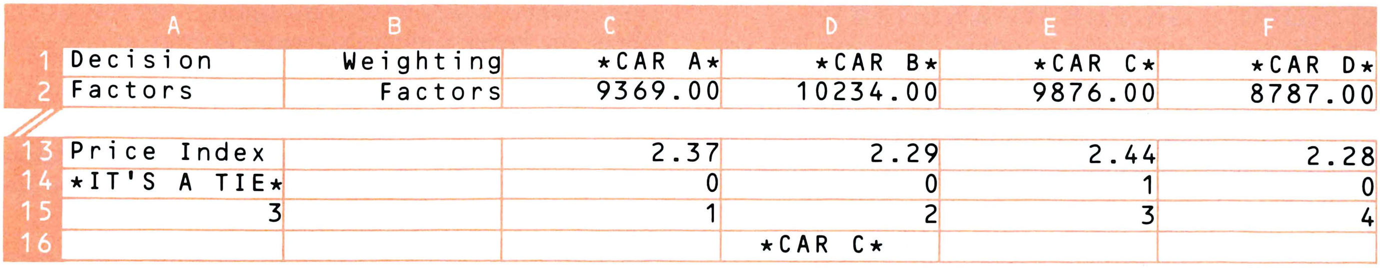
First you need to determine if there is a tie. Since the highest Price Index is indicated in C14 through F14 by the value 1, the sum of these cells equals 1 when only one car has the highest index. If two or more cars are tied for the maximum value, the sum of C14 through F14 is greater than 1.

Use the @IF function to tell the program that if the sum of C14 through F14 equals 1 (no tie), then return the label of the car with the highest Price Index. Because



	A	B	C	D	E	F
1	Decision	Weighting	*CAR A*	*CAR B*	*CAR C*	*CAR D*
2	Factors	Factors	9369.00	10234.00	9876.00	8787.00
10	service rec.	2	6	8	4	5
11	trunk space	1	1	9	2	6
12	----------	----------	----------	----------	----------	----------
13	Price Index		2.37	2.29	2.44	2.28
14					***********	

FIGURE 2. *Because CAR C has the highest Price Index, the cell below its index is filled with asterisks, indicating that this is the car you should buy.*



	A	B	C	D	E	F
1	Decision	Weighting	*CAR A*	*CAR B*	*CAR C*	*CAR D*
2	Factors	Factors	9369.00	10234.00	9876.00	8787.00
13	Price Index		2.37	2.29	2.44	2.28
14	*IT'S A TIE*		0	0	1	0
15	3		1	2	3	4
16				*CAR C*		

FIGURE 3. *The lookup table in rows 14 and 15 is used to find the column number of the car with the highest Price Index. This number is displayed in A15 and is used with the @LCHOOSE function to return in D16 the label of the car you should buy. In the case of a tie, the label from A14 is returned.*

the number that the lookup table returns in A15 corresponds to the car you should buy, use this number with the @LCHOOSE function to select the appropriate label from C1 through F1.

If the @IF statement determines that the sum of C14 through F14 does not equal 1, then the program should return a label indicating a tie. Use the @LABEL function to accomplish this.

Begin by putting the label for a tie in A14:

```
AVO   >A14:"*IT'S A TIE*
```

Next, enter in D16 the @IF statement that returns the proper label:

```
AVO   >D16:@IF(@SUM(C14...F14)=1,
      @LCHOOSE(A15,C1...F1),@LABEL(A14))
```

Because rows 14 and 15 contain information that you do not need to see in order to know which car you should buy, use the Advanced Version's Hide option of the Attribute command to make the contents of these rows invisible. (See Figure 4.) Enter:

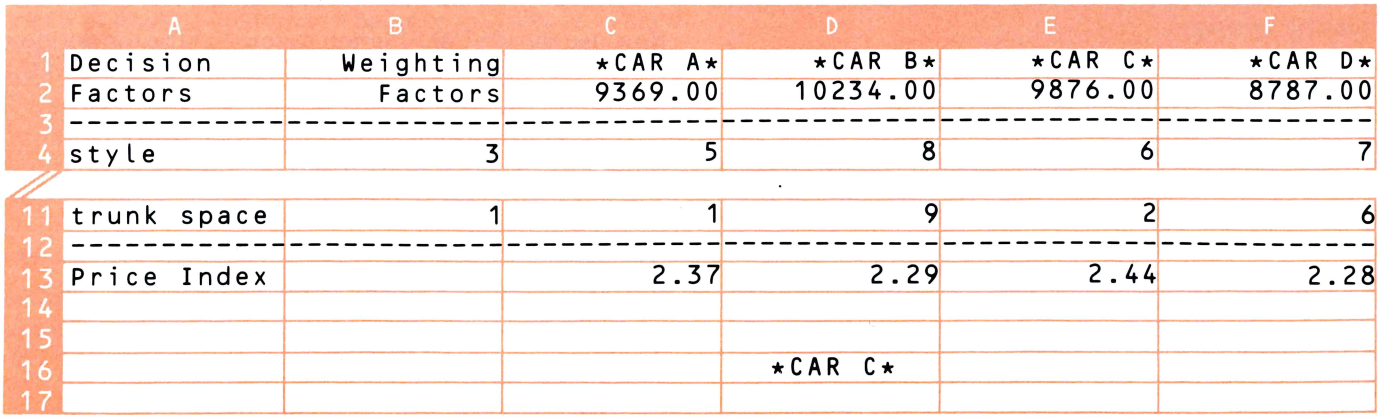
```
AVO   >A14:/AHY
AVO   >A14:/R(A):A14...F15
```

Changing Priorities

Now that you have built the decision analysis model, you can reassess your priorities, make revisions to any of the prices or ratings, and the program will instantly indicate what your new decision should be. For example, perhaps you discover that you can get a $500 discount on CAR B so that it will cost you $9734. When you change the price, the program shows you that you should still buy CAR C. However, if you can talk the dealer down another $200, to $9534, CAR B becomes the best buy, considering the priorities in this model.

What if your old car just broke down? You may decide that the availability of a new car is more important than you previously thought. So you give it a Weighting Factor of 7 and move safety record and acceleration down one place each in the ranking. In this case, you should buy CAR A.

Whether you work with the Advanced Version of the program or with another version, you can use this decision analysis model to evaluate your options quickly and logically and tell what your decision should be. Apply the model to any problem in which you have two or more options to consider and various factors affect your decision. ☐



	A	B	C	D	E	F
1	Decision	Weighting	*CAR A*	*CAR B*	*CAR C*	*CAR D*
2	Factors	Factors	9369.00	10234.00	9876.00	8787.00
3	------------------	------------------	------------------	------------------	------------------	------------------
4	style	3	5	8	6	7
11	trunk space	1	1	9	2	6
12	------------------	------------------	------------------	------------------	------------------	------------------
13	Price Index		2.37	2.29	2.44	2.28
14						
15						
16				*CAR C*		
17						

FIGURE 4. *The contents of rows 14 and 15, which contain the lookup table, are hidden with the /AHY command of the Advanced Version of the program. The label of the car to buy is displayed in D16.*

WINDOWS ON THE WORKSHEET

The VisiCalc® program's Window command lets you do more than just create two windows. For instance, when you use this command, you can set different formats in each window. Each window can also have a different column width and independent title settings.

To see how the Window command can help you manage your worksheet and conveniently locate data, consider the case of a company that we'll call the Colonial Oil Company. Every month, Colonial Oil compiles an Accounts Receivable statement that summarizes its customers' past due, current, and total balances. The company also maintains records of each customer, including such information as address, phone number, oil tank size, type of delivery (automatic or on a "will call" basis), latest delivery date, and credit limit.

If all this information were contained in one window, it would be cumbersome, and focusing on the desired information could be difficult. Therefore, Colonial Oil uses two windows: one for Accounts Receivable and the other for Customer Information.

To set up the Colonial Oil Company model shown in Figure 5, begin by making the columns 12 characters wide and setting the global format to display values in dollars and cents. Type:

```
/GC12
/GF$
```

Row 4 will contain customer names, which we want to align at the right margin. Type:

```
>B4:/FR/R:C4...Z4
```

Because we want the account numbers to be displayed as integers, set a local integer format in row 5 by typing:

```
>B5:/FI/R:C5...Z5
```

Enter the labels in rows 1 and 2, and create a continuous dashed line across row 3. Now enter the labels down column A in rows 4 through 9, as shown in Figure 5.

Because we do not want rows 1 through 3 or column A to disappear from the screen while scrolling through the list of customers, fix both horizontal and vertical titles by typing:

```
>A3:/TB
```

Creating Windows

To set up the second window for the Colonial Oil model, split the screen horizontally at A10:

```
>A10:/WH
```

Next, move the cursor to the bottom window and make the columns 36 characters wide. Enter:

```
/GC36
```

The global dollars-and-cents format set in the top window was transferred into the bottom window when the screen was split. A feature of the VisiCalc program allows you to set different global formats in each window. Since most of the entries in the bottom window will be labels, the general (default) format is more appropriate here. Enter:

```
/GFG
```

You will be fixing different titles in the bottom window, so type **/TN** to remove the title settings that were automatically transferred from the top window.

In column A, rows 10 through 20, enter the labels shown in Figure 5. Also create the dashed line across row 11. All the columns to the right of A can be used to record information about the individual customers, and any additional information that you want to include (such as make of oil burner or service contract in effect) can be entered below row 20.

To set the proper boundaries in the bottom window, move the cursor so that A10 is in the upper left corner.

Now fix both the vertical and horizontal titles by typing:

```
>A11:/TB
```

Note that with /TB set at A11, sending the cursor to any row above 10 will unset the horizontal titles in the bottom window.

Because we will enter the Account Number (row 13) and Credit Limit (row 20) as values, set a local left-aligned format across rows 13 and 20:

```
>B13:/FL/R:C13...Z13
>B20:/FL/R:C20...Z20
```

All other entries in the bottom window will be made as labels.

Simplifying Data Entry

Now you are ready to enter detailed account and customer information. You can simplify some of the data entry by cross-referencing information in the windows and by using the @SUM function.

For example, you don't need to enter the Account Number in both windows. With the cursor in the bottom window, set up row 13 with an expression that references row 5 in the top window. Type:

```
>B13:+B5®/R:C13...Z13:R
```

Now use the @SUM function in row 8 of the top window to calculate the total balances. Move the cursor to the top window (;) and enter:

```
>B8:@SUM(B6...B7)®/R:C8...Z8:RR
```

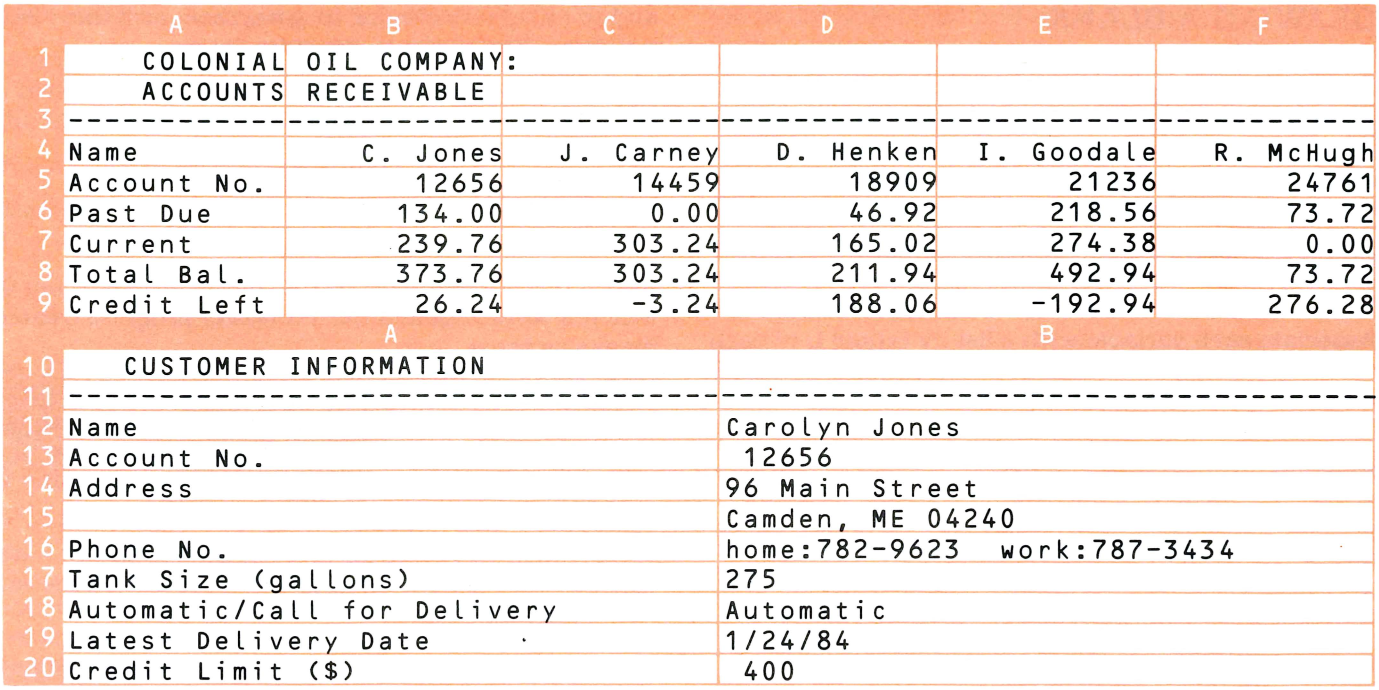
Similarly, you can total all of the customers' Past Due, Current, and Total Balance amounts by entering:

```
>AA6:@SUM(B6...Z6)®/R:AA7...AA8:RR
```

In row 9, enter and replicate the expression that calculates the Credit Left by subtracting the Total Balance (in the top window) from the Credit Limit (in the bottom window) for each customer. Type:

```
>B9:+B20-B8®/R:C9...Z9:RR
```

The Synchronized Scrolling option of the Window command will display the related information for each



	A	B	C	D	E	F
1	COLONIAL	OIL COMPANY:				
2	ACCOUNTS	RECEIVABLE				
3	------------	------------	------------	------------	------------	------------
4	Name	C. Jones	J. Carney	D. Henken	I. Goodale	R. McHugh
5	Account No.	12656	14459	18909	21236	24761
6	Past Due	134.00	0.00	46.92	218.56	73.72
7	Current	239.76	303.24	165.02	274.38	0.00
8	Total Bal.	373.76	303.24	211.94	492.94	73.72
9	Credit Left	26.24	-3.24	188.06	-192.94	276.28

	A	B
10	CUSTOMER INFORMATION	
11	------------------------------------	------------------------------------
12	Name	Carolyn Jones
13	Account No.	12656
14	Address	96 Main Street
15		Camden, ME 04240
16	Phone No.	home:782-9623 work:787-3434
17	Tank Size (gallons)	275
18	Automatic/Call for Delivery	Automatic
19	Latest Delivery Date	1/24/84
20	Credit Limit ($)	400

FIGURE 5. *By scrolling across the columns in the top window, you can display the Colonial Oil Company's Accounts Receivable. The related Customer Information for each account will appear in the bottom window. Both horizontal and vertical titles are fixed at A3 in the top window and at A11 in the bottom window. In addition, the top window uses the global dollars-and-cents format, a local right-aligned format in row 4, and a local integer format in row 5. The bottom window uses the general (default) format globally, and rows 13 and 20 have a local left-aligned format. Notice that the column width is 12 in the top window and 36 in the bottom window.*

customer in both windows simultaneously. You can enter this command regardless of where the cursor is located. Type:

```
/WS
```

Now enter the rest of the account information in rows 4 through 7 of the top window and in rows 12 and 14 through 20 of the bottom window.

The View from Both Windows

Whenever the Colonial Oil Company's accountants need to know a customer's balance and where to send the bill, they can scroll through the top window to find the amount, and the bottom window will display the address. However, they don't have to look at individual Customer Information records to determine the total amount owed at the end of the month, since this data is all contained in the Accounts Receivable statement.

The manager of the service department has to look only at the delivery information in the Customer Information window to determine if an oil delivery should be scheduled. Before authorizing deliveries, however, the manager must check whether customers have exceeded their credit limit, as indicated in the Credit Left row in the top window.

By using two windows when you set up a model, you can separate your information to make it more manageable and accessible. Essentially, you can deal with your data independently in each window, setting different titles, column widths, and global formats. At the same time, you can display related information in the two windows by using synchronized or unsynchronized scrolling to suit your needs. ☐

BACK ISSUES AVAILABLE

Back issues of SATN are now available. The cost is $7.00 per copy for orders mailed within the USA. If your order is to be mailed outside the USA, add $2.50 for shipping and handling to the full price of the order.

To order back issues, please send your name and address, a list of the specific issues of SATN you want (by Volume and Issue numbers), and a check or money order payable to SATN to:

SATN Back Issues
P.O. Box 99
Newton, MA 02162

DRAWING A BLANK

When solving for a list of values, you will get blanks in your lists if, for some elements, the TK!Solver® program either does not have enough information to solve for an output value or an error occurs. TK!Solver will return no value in the affected output lists, leaving a blank, and will proceed to the next element.

Here's a simple example. Enter the following rule:

```
c = SQRT(a-b)
```

Associate each variable with a list by typing **L** in each variable's Status field. Make *a* and *b* input variables by also typing **I** in their Status fields. Dive to the subsheet for list *a* and enter these values:

Element	**Value**
1	26
2	8
3	1
4	11

Go to the subsheet for list *b* and enter:

Element	**Value**
1	1
2	4
3	3
4	2

Solve for all the list values by typing **/L!**. The results are shown in Figure 6.

At each step, the TK!Solver program evaluated the square root of the difference between *a* and *b* and returned that value for *c*. At the first element, *c* equals SQRT(26–1), which is 5. At the second element, *c* equals SQRT(8–4), or 2. At the third element, however, *c* evaluates to SQRT(1–3), or SQRT(–2). Negative numbers can't have square roots, so the TK!Solver program signals an error by beeping and leaves a blank in list *c*. The program then proceeds to the next set of input values and returns the final value for *c*.

The TK!Solver program treats blanks in lists as holes, with no defined value. The VisiCalc® program (and some other spreadsheets) evaluates blanks to zero.

If TK!Solver is asked to use a blank value, it marks an error. To see this, add the following rule to your model:

```
d = ELEMENT('c,3)
```

That rule says that *d* has the value of the third element in list *c*. Now solve by typing **!**. The TK!Solver program beeps and displays an error message. If you move the cursor to the Status field next to the new rule, the message > **ELEMENT: Argument error** will show on the Prompt/Error line. The program is unable to evaluate *d*, since the blank in the third position has no defined value. An error is also indicated in the Status field for variable *d*.

List Functions and Blanks

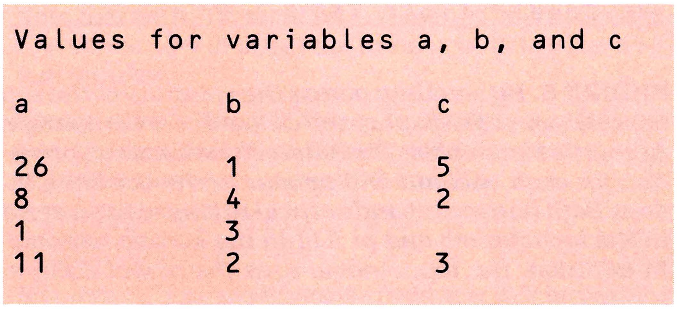
Built-in functions that can use lists (COUNT, MAX, MIN, and so on) ignore blanks, treating them as if they were not there. For example, consider list *c*:

Element	**Value**
1	5
2	2
	(blank)
4	3

Here, COUNT('*c*) equals 3, not 4. MIN('*c*) equals 2, not 0, which it would equal if the TK!Solver program treated blanks as zeros.

The distinction between a blank and a zero is particularly significant in two other built-in functions that use lists, NPV and POLY. In these functions, the position of a value in a list is directly related to how that value is treated in calculations.

We'll look at the behavior of these functions when the lists involved contain blanks. First, reset the program by typing **/RAY**.



```
Values for variables a, b, and c

a           b           c

26          1           5
8           4           2
1           3
11          2           3
```

FIGURE 6. *At the third element, the value of c was* SQRT*(1–3), or* SQRT*(–2). Negative numbers don't have square roots, so an error was signaled and a hole left in list c.*

NPV

In using NPV to evaluate an investment, the income from the investment can be entered in a list. The initial outlay would be the first value in the list, and the future cash flow would make up the rest of the list.

Enter this rule:

```
WORTH = NPV(rate,'cashflow)
```

The variable *rate* is the discount, or interest, rate; the cash flow values will be entered in the list *cashflow*.

Go to the List Sheet, create the list *cashflow*, dive to its subsheet, and enter the following two values:

Element	**Value**
1	-100
2	110

Those values indicate that you are paying out $100 now and will receive $110 in one year's time. Return to the Variable Sheet and give *rate* the value of **.1**, for 10%. Now solve. The net present value is 0, as expected: $110 in one year's time has the same value that

$100 now has, assuming 10% interest.

Now return to the List subsheet for *cashflow* and insert a blank in the list by putting the cursor on the row for the second element and typing **/I**.

You now have the following list:

Element	**Value**
1	-100
	(blank)
3	110

We added the blank artificially, but such a blank could have arisen naturally if at some point you solved for the cash flow amounts and there was not enough information for the second element or an error occurred.

Solve the model again by typing **/!**. The net present value is still 0, since the second element, a blank, is simply ignored. Therefore, 110, even though positioned on the subsheet as element 3, is actually considered as the second element of the list, that is, the cash flow at one year.

If the investment is such that the first year's income is 0 and the second year's is $110, you have to say that explicitly. That is, you must change the blank to a 0 by typing **0** in the Value field for the second element, creating the following list:

Element	**Value**
1	-100
2	0
3	110

Now when you solve the model it shows the net present value is –9.09.

In NPV, then, the position of a value in the cash flow list is crucial because it indicates the time of the payment. A blank in the list will "pull up" the values following it, causing them to reflect a payment at an earlier time and changing the way they are calculated by the NPV function.

The position of a value in a list is also vital in the TK!Solver program's POLY function.

POLY

POLY returns the value of a specified polynomial. The coefficients for the polynomial can be entered in a list. If so, POLY has the syntax POLY(*x*, '*list*), where *x* is the number whose polynomial value is to be calculated.

Go to the List Sheet and create a list *e*. Now dive to the subsheet and enter the following values, which represent the coefficients of a polynomial:

Element	**Value**
1	2
2	3
3	4

That list corresponds to the polynomial $2x^2 + 3x + 4$.

Go to the Rule Sheet and enter the rule:

```
y = POLY(3,'e)
```

That is, *y* is the value of the given polynomial when *x* equals 3.

Solve. POLY(3,'*e*) has the value 31 ($2 \times 3^2 + 3 \times 3 + 4 = 31$).

Now create a blank element in list *e* by going to the List subsheet, putting the cursor on the row for the second element, and typing **/I**. The list now reads:

Element	**Value**
1	2
	(blank)
3	3
4	4

Solve. POLY(3,'*e*) still evaluates to 31, since the blank is treated as if it weren't there.

Change the blank to a 0 by typing **0** in the Value field for the second element, creating the following list:

Element	**Value**
1	2
2	0
3	3
4	4

Now when you solve, POLY(3,'*e*) equals 67. The zero in the list indicates that the coefficient for the corresponding term in the polynomial is 0. In this case, the zero applies to the x-squared term. So the polynomial is now $2x^3 + (0x^2) + 3x + 4$, not $2x^2 + 3x + 4$ as it was before.

With POLY, then, the position of a value in the list of coefficients indicates to which term in the polynomial the value refers. The last value in the list is the coefficient of the "zeroeth-order" term (x^0), the next to last value is the coefficient of the first-order term (x^1), and so on. A value above a blank in a list is treated as if it were the coefficient for a lower-order term than was probably intended. That changes the evaluation of the polynomial.

So be careful if you are used to programs that treat blanks as zeros. The TK!Solver program treats blanks as blanks, with no defined value. ☐

HAVE YOU MOVED?

If you have, let us know. To keep your issues of SATN coming to you regularly, we need to know your current address. Send a card or letter, indicating both your old address and your new one, to:

SATN Changes
P.O. Box 99
Newton, MA 02162

Remember to allow about 4 weeks for the change to take effect.

FROM OUR READERS

I'm sure that you prefer models with no forward references, unless it's impossible to solve the problem without them.

In your pointer on expanding an @LOOKUP range (Vol. III, No. 2), forward references are used. You mention that when extending an @LOOKUP range down additional columns, recalculations are necessary to return the correct number if the index value is in the last column.

Expanding the range of an @LOOKUP function could be done without any forward references by changing the sequence of the columns used. For instance, enter:

```
>C1:1
>D1:10*C1
>C2:1+C1
>D2:10*C2
>C2:/R...D2:C3...C253:RR
```

Now make the bridge in C254 and D254:

```
>C254:253.1
>D254:@LOOKUP(G1,A1...A254)
```

Next, enter the second table in columns A and B:

```
>A1:254
>B1:10*A1
>A2:1+A1
>B2:10*A2
>A2:/R...B2:A3...A254:RR
```

As in your pointer, use G1 for the index value and the corresponding @LOOKUP value will be returned in G2. Enter:

```
>G2:@LOOKUP(G1,C1...C254)
```

If you now enter any value between 1 and 508 in G1, one recalculation will return the correct value in G2.

If you want to further expand the @LOOKUP range, simply insert two columns at column A, make the bridge from the new columns C and D to the new columns A and B, and again one recalculation solves the problem!

Gerrit Horlings
Amsterdam
The Netherlands

Editor's Note: If you are using the Advanced Version of the program on an IBM PC, set the global order of recalculation to column (/GOC). The default, natural order, sometimes requires recalculation if your @LOOKUP function refers to an expression.

Another way to expand your @LOOKUP range without using forward references is by using the @IF function. For example, in the model we gave in our pointer (if you have Gerrit's model on your screen, you can re-create our model by moving columns C and D to the left of columns A and B), remove the bridge in row 254 by entering:

```
>A254:/B
>A254:/R:B254...D254
```

Now enter the following expression in G2:

```
>G2:@IF(G1<254,@LOOKUP(G1,A1...A253),
 @LOOKUP(G1,C1...C253))
```

This expression tells the VisiCalc program to search column A if the index number is less than 254. Otherwise, search column C.

By building an @IF function within another @IF function, you can expand your @LOOKUP range to include as many columns as you like, without ever using a forward reference. For instance, if columns A, C, and E contained index numbers, you would use the expression:

```
@IF(G1<254,@LOOKUP(G1,A1...A253),
 @IF(G1<507,@LOOKUP(G1,C1...C253),
  @LOOKUP(G1,E1...E253))
```

RELATED PRODUCTS AND PUBLICATIONS CHECKLIST

No one at Software Arts has reviewed or examined any of these products to determine whether they perform as claimed or as listed below. An inclusion in this listing does not constitute an endorsement by Software Arts of that product. We also wish to point out that if any of the products listed alter, patch, or copy the VisiCalc® program, they could be in violation of your user licensing agreement and/or the copyright.

Since we have not reviewed any of the products, we cannot provide you with any more information than is contained in the list. If you want to learn more or you wish to purchase any of the products listed, contact your dealer first. Although we have identified distributors in some listings, many distributors do not accept orders from individuals for single copies of software or pieces of hardware.

SOFTWARE

Only software that claims to complement the VisiCalc program will be listed.

Templates

Bottomline-V

- Runs on Apple II+, Apple IIe, Apple III, IBM PC/XT, and TRS-80 Models II, III, IV, and 16
- Requires 64K

Contains several integrated financial statements, including budget, income statement, balance sheet, sources and uses of cash, and financial-ratio analysis. Provides up to five-year forecasts on investment utilization, company value, profitability measures, and cash flow and budget requirements.

Distributed by Strategic Software Systems, Inc., 1300 Dove St., Suite 200, Newport Beach, CA 92660

Suggested retail price: $300.00

Calc-Plus, Full System

- Runs on Apple II, Apple II + , and Apple IIe
- Requires 64K and two disk drives

Consists of a group of accounting templates designed to handle Payroll, Accounts Receivable, Accounts Payable, Depreciation, Cash Disbursement, and a Cash Receipts Journal. Data may be drawn from each module to construct a balance sheet and an income statement. Each module includes audio instructions and online documentation.

Distributed by Micro Instructional, 3453 N.W. 55th St., Fort Lauderdale, FL 33309

Suggested retail price: $169.95

Construction Estimating Templates

- Runs on Apple II, Apple II + , Apple IIe, Atari 800 series, Hewlett Packard 125, IBM PC/XT, and TRS-80 Models I, II, III, IV, and 16
- Requires 64K

An 18-template program that enables general contractors and architects to estimate cost of building a home, allowing for contingencies, overhead, and profit margin.

Distributed by Software Models, P.O. Box 1029, Crestline, CA 92325

Suggested retail price: $59.95

Entrepreneur Software Series

- Runs on Apple II, Apple II + , Apple IIe, Apple III, IBM PC/XT, and TRS-80 Models I, II, III, IV, and 16
- Supports DIF format

Consists of custom VisiCalc templates and DIF files for use with Entrepreneur business-planning manuals. Six diskettes, each with multiple templates, are included: Business Services, Consumer Services, Food & Beverage Services, Facilities Oriented Businesses, Retail, and Low Investment.

Distributed by Entrepreneur, 2311 Pontius Ave., Los Angeles, CA 90064

Suggested retail price: $200.00

Finance Projections

- Runs on Apple II, Apple II + , Apple IIe, IBM PC/XT, and TRS-80 Models II, III, IV, and 16
- Requires 64K

A collection of spreadsheet templates for small businesses to create cash budget projections for loans and to facilitate periodic planning of a cash budget. Can be used to prepare break-even analyses, *pro forma* balance sheets, and income sheets. Depreciation tables, financial ratio formulas, and net present values are included.

Distributed by Software Models, P.O. Box 1029, Crestline, CA 90064

Suggested retail price: $59.95

Communications

Apple-IBM PC File Transfer Program

- Runs on Apple II, Apple II + , Apple IIe, and IBM PC/XT
- Requires an Apple serial card

Allows the transfer of files from an Apple II, II + , or IIe to an IBM PC or IBM XT. The product includes a communications program for the IBM PC and an adapter for connecting the two systems. Supports transfer of random, text, data, and binary files, as well as VisiCalc data. High level language programs written in BASIC, Pascal, Fortran, and Cobol can also be transferred.

Distributed by Personal Computer Products, 1400 Coleman Ave., Suite C-18, Santa Clara, CA 95050

Suggested retail price: $94.95

DOS File Exchange II (DFXII)

- Runs on Apple II
- Requires a modem

Uses a modem to transfer standard Apple DOS 3.3 files from one Apple II to another Apple II. Two users can view the same VisiCalc worksheet and enter data simultaneously. The save command saves the file on both systems.

Distributed by Alpha Logic Business Systems, Inc., 4119 North Union Rd., Woodstock, IL 60098

Suggested retail price: $79.95

Other Software

Calc-Kit

- Runs on Apple II + , Apple IIe, and IBM PC
- Requires 48K (64K recommended) and a dot-matrix printer with graphics interface for printing graphics

Adds the following features to the VisiCalc program: data management, graphics, documentation, and report generation systems. Is menu-driven and offers two levels of instruction. A 46-page user manual is included.

Distributed by El Dorado Software, Inc., 549A Castro St., San Francisco, CA 94114

Suggested retail price: $100.00

Profin

- Runs on IBM PC/XT and TRS-80 Model II
- Requires 64K memory
- Supports DIF format

A menu-driven financial analysis program that helps users evaluate criteria for investment decisions. Reports on investment returns, discounted cash flows, and capital expenditures. Completes a personalized balance sheet.

Distributed by Business Software Pty., Ltd., Chatswood, Australia

Suggested retail price: $295.00

PUBLICATIONS

Regular Columns

Business Computing, "Spreadsheets," by Penny Alexander

December 1983: "VisiCalc Companions" describes add-on products that enhance the VisiCalc program.

January 1984: "Keeping Secrets with VisiCalc" instructs VisiCalc users on how to conceal screen data with symbols that can be switched on and off.

February 1984: "Keeping a Mini-Ledger" shows how to build a Check Register and Mini-General Ledger with the VisiCalc program.

March 1984: "Tax Templates Prove Taxing" explores the advantages and disadvantages of using prewritten tax templates. The differences between spreadsheet templates that are tax return planners and those that are tax return preparers are also explained.

Desktop Computing, "Visulating," by Barry Bayer

August 1983: Using the VisiCalc program in a hospital pharmacy to aid in mixing solutions.

September 1983: Using the VisiCalc program to prepare loan applications; how to accelerate the operating speed of your spreadsheet program; VisiCalc hints for amateur radio operators.

October 1983: Using the VisiCalc program to prepare college financial aid forms easily.

November 1983: Looks at the running speeds of various spreadsheet programs on the Compaq computer.

December 1983: Examines some VisiCalc tax templates that make tax preparation easier.

Softalk, "Ventures with VisiCalc," by Joe Shelton

August 1983: Examines the features of VisiCalc Advanced Version for the Apple IIe.

September 1983: Creates a template for calculating the cost of borrowing money.

October 1983: Modifies the template introduced in Shelton's September column to take advantage of some of the features of VisiCalc Advanced Version.

Other Publications

Mentor

- Runs on IBM PC
- Requires 64K and one double-sided disk drive

An online computer magazine concentrating on application routines for VisiCalc and other programs. Consists of both articles in standard magazine format and self-running demos of products from manufacturers. Introductions and documentation for the demos are included.

Distributed by Mentor Computer Services, 1362 Pacific Ave., Santa Cruz, CA 95061

Suggested retail price: $99.00 for six issues; $19.97 for a single issue

VisiCalc
Advanced Version
Software Arts

VISICALC® ADVANCED VERSION NOW ONLY $99.95

Software Arts™ is making this special upgrade offer to IBM® PC, XT and Apple® //e owners for a limited time only.

VisiCalc® Advanced Version is a state-of-the-art spreadsheet for planning, budgeting, forecasting and other types of complex projections and calculations.

It offers advanced capabilities including:

- Variable Column Widths
- Expanded Formatting Options
- Flexible Printer Control
- Financial Functions including Net Present Value and Internal Rate of Return
- A "Smart" Help Facility
- Microsoft® Mouse support*
- Full Word Prompts*
- Keystroke Memory™ for executing several commands with a single keystroke.

And it's fully compatible with all current VisiCalc files.

What you get for the price. The $99.95 package comes with a full program disk. A 40-page reference guide. A product information booklet that explains all the advanced features. Full license, warranty and customer support plan. Backed with our technical support.

System requirements: IBM PC, XT or IBM compatible with 192K memory; DOS 1.1 or DOS 2.0; two double-sided disk drives (IBM PC) or one double-sided disk drive and one hard disk (IBM XT). Apple //e; Apple //e Extended 80-column Text card (128K memory); DOS 3.3; 2 disk drives.

*IBM version only.

To order call 1-800-CALL800.

For further information call Cathy Brown at 617-431-6600.

Software Arts™

THE CREATORS OF VISICALC,® THE LARGEST SELLING BUSINESS SOFTWARE IN THE WORLD.

Copyright © 1984 Software Arts, Inc. All rights reserved.

To order by mail, complete coupon and mail to: Software Arts, Inc., P.O. Box 638, Newton, MA 02162.

NAME

ADDRESS (NO POSTAL BOXES)

CITY STATE ZIP

☐ Check enclosed (make payable to *Software Arts, Inc.*)

☐ MasterCard ☐ Visa

CREDIT CARD NUMBER EXPIRATION MONTH & YEAR

	Quantity		Total
VisiCalc® Advanced Version for IBM	____	@ $99.95	$ ____
VisiCalc® Advanced Version for Apple //e	____	@ $99.95	$ ____
Mass. residents add 5% sales tax			____
Shipping & Handling			3.00
Total Amount			$ ____